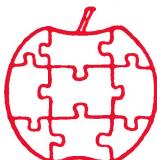


Apple



Assembly

Line

\$1.80

Volume 4 -- Issue 6

March, 1984

In This Issue...

Fast Garbage Collection.	2
Changing VERIFY to DISPLAY	13
Changing Tab Stops in the 68000 Cross Assembler.	15
S-C Macro and GPLE.LC on the //e	16
Redundancy in Tables for Faster Lookups.	17
Speaking of Locksmith.	19
Lancaster's OBJ.APWRT][F	19
About Disk Drive Pressure Pads	20
Will ProDOS Work on a Franklin?.	20
Rod's Color Pattern in 6502 Code	21
Will ProDOS Really Fly?.	28

For some time now we have been selling our S-C Word Processor, complete with all source code on disk. We hoped that some users would send us their improvements, and sure enough they have. Bob Gardner recently sent us a bunch, and that motivated me to go back over the package.

The disk now includes both II/IIPlus and //e versions. The //e version allows an 80-column preview (still only 40-column during edit mode). I added titles and page numbers, a single sheet mode, and more. Even with all the new features, the new object code is a little shorter than the old, leaving even more room for your own modifications and enhancements. I improved the internal documentation. The "manual-ette" is now 10 pages rather than 6. A small tutorial file helps you get started.

The price is still only \$50. Owners of the old version can get a new copy for only \$5.

**Fast Garbage Collection.....Col. Paul Shetler, MD
Honolulu, Hawaii**

When Applesoft programs manipulate strings, memory gradually fills up with little bits and pieces of old strings. Eventually this space needs to be recovered so the program can continue. The process of searching through all the still active strings, moving them back to the top of free memory, and making the remaining space available again is called "garbage collection".

Applesoft will automatically collect garbage when memory fills up. However, the garbage collector in the Applesoft ROMs is pitifully slow. Worse yet, the time to collect is proportional to the square of the number of strings in use. That is, if you have 100 active strings it will take four times as long to collect garbage as if you had only 50 active strings.

Cornelis Bongers, of Erasmus University in Rotterdam, Netherlands, published a brilliant Garbage Collector for Applesoft strings in Micro, August 1982. The speed of his program, when compared to the one residing in ROM, is incredible. And the time is directly proportional to the number of strings, rather than the square of the number of strings. The only problem with his program is that it belongs to the magazine that published it. Or worse yet, it is tied to a program called Ampersoft, marketed by Microsparc (publishers of Nibble magazine) for \$50. When I asked them about a license, they wanted big bucks.

So, I decided to write my own garbage collector, based on the ideas behind Cornelis Bongers' program. And then I further decided to make it available to all readers of Apple Assembly Line, where I myself have received so much help.

There are several catches. Normal Applesoft programs save all string data with the high-order bit of each byte zero (positive ASCII). Further, normal Applesoft programs never allow more than one string variable to point to the same exact memory copy of the string. The method of garbage collection my program uses (Bongers' method) DEPENDS on these constraints. If either is not true, LOOK OUT! Of course, if your Applesoft programs are normal, you need have no fear. Only if you are doing exotic things with your own machine language appendages to Applesoft might these constraints be violated.

The basic concept is fairly simple. Applesoft uses descriptors to point to the string in the string pool. The descriptor consists of three bytes -- the length, and the address of the characters in the string pool.

Strings build down from the top of memory (HIMEM) and the descriptors build up from the end of the program in the variable space. Since a new value assigned to a string is added to the bottom of the string pool, the pool is soon full of "garbage".

Applesoft frees the garbage one string at a time. This

S-C Macro Assembler Version 1.0.....	\$80.00
S-C Macro Assembler Version 1.1 Update.....	\$12.50
Full Screen Editor for S-C Macro Assembler.....	\$49.00
Includes complete source code.	
S-C Cross Reference Utility.....	\$20.00
S-C Cross Reference Utility with Complete Source Code.....	\$50.00
DISASM Dis-Assembler (RAK-Ware).....	\$30.00
Quick-Trace (Anthro-Digital).....	(reg. \$50.00) \$45.00
The Visible Computer: 6502 (Software Masters).....	(reg. \$50.00) \$45.00
 S-C Word Processor (the one we use!).....	\$50.00
With fully commented source code.	
Applesoft Source Code on Disk.....	\$50.00
Very heavily commented. Requires Applesoft and S-C Assembler.	
ES-CAPE: Extended S-C Applesoft Program Editor.....	\$60.00
 AAL Quarterly Disks.....	each \$15.00
Each disk contains all the source code from three issues of "Apple Assembly Line", to save you lots of typing and testing time.	
QD#1: Oct-Dec 1980 QD#2: Jan-Mar 1981 QD#3: Apr-Jun 1981	
QD#4: Jul-Sep 1981 QD#5: Oct-Dec 1981 QD#6: Jan-Mar 1982	
QD#7: Apr-Jun 1982 QD#8: Jul-Sep 1982 QD#9: Oct-Dec 1982	
QD#10: Jan-Mar 1983 QD#11: Apr-Jun 1983 QD#12: Jul-Sep 1983	
QD#13: Oct-Dec 1983 QD#14: Jan-Mar 1984	
 Double Precision Floating Point for Applesoft.....	\$50.00
Provides 21-digit precision for Applesoft programs.	
Includes sample Applesoft subroutines for standard math functions.	
Amper-Magic (Anthro-Digital).....	(reg. \$75.00) \$67.50
Amper-Magic Volume 2 (Anthro-Digital).....	(reg. \$35.00) \$30.00
Routine Machine (Southwestern Data Systems).....	(reg. \$64.95) \$60.00
FLASH! Integer BASIC Compiler (Laumer Research).....	\$79.00
Fontrix (Data Transforms).....	\$75.00
Aztec C Compiler System (Manx Software).....	(reg. \$199.00) \$180.00
IACcalc Spreadsheet Program.....	(reg. \$84.95) \$75.00
The one we use every day. It's better than Visicalc!	
 Blank Diskettes.....	package of 20 for \$45.00
(Premium quality, single-sided, double density, with hub rings)	
Vinyl disk pages, 6"x8.5", hold one disk each.....	10 for \$6.00
Diskette Mailing Protectors.....	10-99: 40 cents each
	100 or more: 25 cents each
Cardboard folders designed to fit into 6"x9" Envelopes.	
Envelopes for Diskette Mailers.....	5 cents each
ZIF Game Socket Extender.....	\$20.00
 Buffered Grappler+ Interface and 16K Buffer.....	(\$239.00) \$200.00
quikLoader EPROM Card.....	(\$179.50) \$170.00
 Books, Books, Books.....	compare our discount prices!
"The Apple II Circuit Description", Gayler.....	(\$22.95) \$21.00
"Understanding the Apple II", Sather.....	(\$22.95) \$21.00
"Enhancing Your Apple II, vol. 1", Lancaster.....	(\$17.95) \$17.00
"Incredible Secret Money Machine", Lancaster.....	(\$7.95) \$7.50
"Beneath Apple DOS", Worth & Lechner.....	(\$19.95) \$18.00
"Bag of Tricks", Worth & Lechner, with diskette.....	(\$39.95) \$36.00
"Assembly Lines: The Book", Roger Wagner.....	(\$19.95) \$18.00
"What's Where in the Apple", Second Edition.....	(\$24.95) \$23.00
"What's Where Guide" (updates first edition).....	(\$9.95) \$9.00
"6502 Assembly Language Programming", Leventhal.....	(\$18.95) \$18.00
"6502 Subroutines", Leventhal.....	(\$17.95) \$17.00

Add \$1.50 per book for US shipping. Foreign orders add postage needed.

*** S-C SOFTWARE, P. O. BOX 280300, Dallas, TX 75228 ***
 *** (214) 324-2050 ***
 *** We accept Master Card, VISA and American Express ***

n-square method takes forever, when there are large string arrays. Bongers introduced the idea of marking active strings in the pool by setting the third byte in the string to a negative ASCII value, then storing the location of the descriptor in the first two bytes. The first two bytes of the string are saved safely in the address of field of the descriptor. The address previously in the address field will be changed anyway after all the strings are moved up in memory.

Another pass through the string pool moves all active strings as high in memory as they can go, retrieves the first two characters from storage in the descriptor, and points it to the new string location.

Since three bytes are used in the active strings, one and two character strings require different treatment. On the first pass through the variable space, the characters pointed to by the 'short' descriptors are stored in the length and, if len=2, the low address byte of the descriptor. The short descriptor is flagged with one or more "FF"'s, since no string can have an address greater than \$FF00.

If short strings are found on the first pass, a third pass returns them to the string pool and points the descriptors to their new location.

Short strings do slow collection a little, however, the number of passes is proportional to the number of strings, and not the number squared.

Bongers' program was driven by calls via the &-statement. Mine differs in that it invoked with the USR function. Although it is easily converted to an ampersand routine, I wrote it using the USR function to provide fast garbage collection with Hayden's compiler (which also uses string descriptors and a string pool). The compiler allows USR functions, but makes & difficult. Another reason is to investigate some uses for USR.

USR(#) converts '#' to a floating point value in the FAC (floating point accumulator) and then jumps via \$0A to the address pointed to in \$0B, \$0C. The results of the machine language subroutine can be returned in the FAC. The USR function, floating point calls, and addresses are described in Apple's BASIC REFERENCE MANUAL FOR APPLESOFT (Product #A2L0006).

The USR argument for my garbage collector requires a number in the range of +32767 to -32767. If the number is negative, the string pool is checked for negative ASCII. If any such characters are found, USR(-1) will return a value of 0, and no garbage collection will be attempted. If no negative ASCII characters are found, garbage collection will proceed. In this case USR(-1) returns the number of bytes of free space after collection.

If the USR argument is zero, for example K = USR(0), then collection is forced and USR will return the amount of free space. This is slightly faster than calling with USR(-1),

because the preliminary scan for negative ASCII bytes is skipped. But USR(-1) is safer, if you are not sure.

If you use a positive argument N in the USR function, then no garbage collection will be performed unless there is less than 256*N bytes of free space left. Whether or not collection is performed, USR will tell you how much free space is left.

Only the lower five bits of the USR argument are tested. This means that USR(32) is the same as USR(0), USR(33) is the same as USR(1), and so on.

I have shown the program as residing at \$9400, but of course you may re-assemble it for any favorite place.

The following Applesoft program makes a lot of garbage, and sees to the collection of it using my garbage collector. If the call to the USR function in line 245 left out, the program dies for 47 seconds while Applesoft does its own garbage collection. With the USR call as shown, the delay is less than one second.

```
100 HIMEM: 37888: REM $9400
110 PRINT CHR$(4)"BLOAD B.FAST GARBAGE COLLECTOR"
120 POKE 10,76: POKE 11,0: POKE 12,148
120 N = 25: DIM A$(N,N),B$(N,N)
210 FOR I = 1 TO N: FOR J = 1 TO N
220 A$(I,J) = "X": B$(I,J) = "Y": NEXT : NEXT
230 FOR I = 1 TO N
240 F = ( PEEK (112) * 256 + PEEK (111)) - ( PEEK (110) * 256 + PEEK (1
09)): PRINT F"; ";
245 L = USR (2)
250 FOR J = 1 TO N
260 FOR K = 1 TO 10:A$(I,J) = A$(I,J) + "": NEXT
270 PRINT "": NEXT
280 PRINT : NEXT
```

```
1000 *SAVE S.FAST GARBAGE COLLECTOR
1010 -----
1020 *   FAST GARBAGE COLLECTOR
1030 -----
1040 *   BY COL. PAUL SHETLER, MD
1050 *   INSPIRED BY CORNELIS BONGERS
1060 -----
1070 *
1080 *   CALL FROM APPLESOFT WITH K=USR(N)
1090 *
1100 *   IF N=0, THEN COLLECTION FORCED
1110 *
1120 *   IF N<0, THEN POOL CHECKED FOR NEG ASCII.
1130 *       IF NO NEG ASCII, THEN GC FORCED
1140 *       IF NEG ASCII FOUND, THEN
1150 *           SET USER(#)=0 AND QUIT.
1160 *
1170 *   IF N>0, THEN COLLECTION PERFORMED ONLY IF
1180 *       LESS THAN N*256 BYTES OF FREE
1190 *       SPACE LEFT.
1200 -----
1210 *   THE APPLESOFT PROGRAM MUST INCLUDE
1220 *   THE FOLLOWING STATEMENTS TO SET UP
1230 *   THIS GARBAGE COLLECTOR:
1240 *
1250 *   100 HIMEM:37888:REM$9400
1260 *   110 PRINT CHR$(4)"BLOAD B.FAST GARBAGE COLL
1270 *       ECUTOR"
1280 *   120 POKE 10,76 : POKE 11,0 : POKE 12,148
```

```

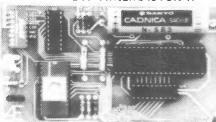
1290 *
1300 * EQUATES FOR GARBAGE COLLECTION
1310 *
0006- 1320 SHORT.FLAG .EQ $06
0007- 1330 STRING.LENGTH .EQ $07
0019- 1340 INDEX .EQ $19
001B- 1350 OFFSET .EQ $1B
001D- 1360 ARRAY.END .EQ $1D
1370 *
1380 * USER(#) EQUATES
1390 *
00A0- 1400 FACMO .EQ $A0
00A1- 1410 FACLO .EQ $A1
E10C- 1420 AYINT .EQ $E10C
E2F2- 1430 GIVAYF .EQ $E2F2
1440 *
1450 * STANDARD EQUATES
1460 *
009B- 1470 LOWTR .EQ $9B
0008- 1480 FORPNT .EQ $08
006D- 1490 STREND .EQ $6D
0069- 1500 VARTAB .EQ $69
0071- 1510 FRESPC .EQ $71
006F- 1520 FRETOP .EQ $6F
0073- 1530 MEMSIZE .EQ $73
006B- 1540 ARYTAB .EQ $6B
1550 *
1560 .OR $9400
1570 .TF B.FAST GARBAGE COLLECTOR
1580 *
1590 USR.GARBAGE.COLLECTOR
9400- 20 OC E1 1600 JSR AYINT CONVERT USR ARGUMENT TO INTEGER
1610 * WITH HIBYTE IN FACMO, LO IN FACLO
9403- A5 A0 1620 LDA FACMO IS # MINUS?
9405- 30 1A 1630 BMI .3 ...NEED TO CHECK FOR NEG ASCII
9407- A5 A1 1640 LDA FACLO
9409- 29 1F 1650 AND #$1F 8136 BYTES
940B- F0 1F 1660 BEQ .4 ...IF =0 THEN FORCED COLLECTION
940D- 18 1670 CLC
940E- 65 6E 1680 ADC STREND+1
9410- C5 70 1690 CMP FRETOP+1
9412- B0 18 1700 BCS .4 ...NEED TO COLLECT NOW
1710 *---CALC FREE SPACE---
1720 .1 SEC
1730 LDA FRETOP
1740 SBC STREND
1750 TAY LO BYTE
1760 LDA FRETOP+1
1770 SBC STREND+1
1780 *---FLOAT (AY) FOR USR RESULT---
1790 .2 JMP GIVAYF FLOAT (AY) AND RETURN
1800 *---CHECK POOL FOR NEG ASCII---
9421- 20 34 95 1810 .3 JSR SET.STRING.POOL.STROLL
9424- 20 3D 95 1820 JSR FIND.NEXT.NEG.BYTE.IN.POOL
9427- A9 00 1830 LDA #0 PREPARE ZERO IN CASE NEG ASCII
9429- A8 1840 TAY
942A- B0 F2 1850 BCS .2 ...FOUND SOME NEG ASCII
1860 *---COLLECT GARBAGE NOW---
1870 .4 JSR MARK.ALL.ACTIVE STRINGS
942C- 20 4C 94 1870 JSR RAISE.ALL.ACTIVE STRINGS
942F- 20 9D 94 1880 *---FINAL CLEAN UP---
1890 *
9432- A5 9B 1900 LDA LOWTR STORE NEW BOTTOM OF STRING POOL
9434- 85 71 1910 STA FRESPC
9436- A5 9C 1920 LDA LOWTR+1
9438- 85 72 1930 STA FRESPC+1
943A- A5 06 1940 LDA SHORT.FLAG NEED TO FIX SHORT STRINGS?
943C- F0 03 1950 BEQ .5 ...NO, NOT ANY SHORT ONES
943E- 20 F2 94 1960 JSR FIX.SHORT.STRINGS
9441- A5 71 1970 .5 LDA FRESPC SET FRETOP TO TOP OF FREE SPACE
9443- 85 6F 1980 STA FRETOP
9445- A5 72 1990 LDA FRESPC+1
9447- 85 70 2000 STA FRETOP+1
9449- 4C 14 94 2010 JMP .1
2020 *
2030 * MARK ACTIVE STRINGS WITH NEG BYTE
2040 *
2050 MARK.ALL.ACTIVE.STRINGS
944C- A9 00 2060 LDA #0 FLAG->NONE
944E- 85 06 2070 STA SHORT.FLAG
9450- 20 65 95 2080 JSR INITIATE.SEARCH

```

Apple Peripherals Are All We Make

That's Why We're So Good At It!

THE NEW TIMEMASTER II



Automatically date stamps files with PRO-DOS

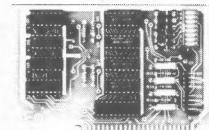
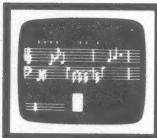
NEW 1984 DESIGN
An official PRO-DOS Clock

- Just plug it in and your programs can read the year, month, date, day, and time to 1 millisecond! The only clock with both year and ms.
- NiCad battery keeps the TIMEMASTER II running for over ten years.
- Full emulation of ALL other clocks. Yes, we emulate Brand A, Brand T, Brand P, Brand C, Brand S and Brand M too! It's easy for the TIMEMASTER to emulate other clocks, we just drop off features. That's why we can emulate others, but others CAN'T emulate us.
- The TIMEMASTER II will automatically emulate the correct clock card for the software you're using. You can also give the TIMEMASTER II a simple command to tell it which clock to emulate (but you'll like the Timemaster mode better). This is great for writing programs for those poor unfortunate that bought some other clock card.
- Basic, Machine Code, CP/M and Pascal software on 2 disks!
- Eight software controlled interrupts so you can execute two programs at the same time (many examples are included).
- On-board timer lets you time any interval up to 48 days long down to the nearest millisecond.

The TIMEMASTER II includes 2 disks with some really fantastic time oriented programs (over 40) including appointment book so you'll never forget to do anything again. Enter your appointments up to a year in advance then forget them. Plus DOS date soit will automatically add the date when disk files are created or modified. The disk is over a \$200.00 value alone—we give the software others sell. All software packages for business, data base management and communications are made to read the TIMEMASTER II. If you want the most powerful and the easiest to use clock for your Apple, you want a TIMEMASTER II.

PRICE \$129.00

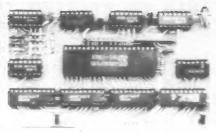
Super Music Synthesizer Improved Hardware and Software



- Complete 16 voice music synthesizer on one card. Just plug it into your Apple, connect the audio cable (supplied) to your stereo, boot the disk supplied and you are ready to input and play songs.
- It's easy to program music with our compose software. You will start right away at inputting your favorite songs. The Hi-Res screen shows what you have entered in standard sheet music format.
- Now with new improved software for the easiest and the fastest music input system available anywhere.
- We give you lots of software. In addition to Compose and Play programs, 2 disks are filled with over 30 songs ready to play.
- Easy to program in Basic, to generate complex sound effects. Now your games can have explosions, phaser zaps, train whistles, death cries. You name it, this card can do it.
- Four white noise generators which are great for sound effects.
- Plays music in true stereo as well as true discrete quadraphonic.
- Full control of attack, volume, decay, sustain and release.
- Will play songs written for ALF synthesizer (ALF software will not take advantage of all our card's features. Their software sounds the same in our synthesizer.)
- Our card will play notes from 30HZ to beyond human hearing.
- Automatic shutdown on power-up or if reset is pushed.
- Many many more features.

PRICE \$159.00

Z-80 PLUS!



- TOTALLY compatible with ALL CP/M software.
- The only Z-80 card with a special 2K "CP/M detector" chip.
- Fully compatible with microsoft disks (no pre-boot required).
- Specifically designed for high speed operation in the Apple IIe (runs just as fast in the II+ and Franklin).
- Runs WORD STAR, dBASE II, COBOL-80, FORTRAN-80, PEACHTREE and ALL other CP/M software with no pre-boot.
- A semi-custom I.C. and a low parts count allows the Z-80 Plus to fly thru CP/M programs at a very low power level. (We use the Z-80A at fast 4MHz.)
- Does EVERYTHING the other Z-80 boards do, plus Z-80 interrupts.

Don't confuse the Z-80 Plus with crude copies of the microsoft card. The Z-80 Plus employs a much more sophisticated and reliable design. With the Z-80 Plus you can access the largest body of software in existence. Two computers in one and the advantages of both, all at an unbelievably low price.

PRICE \$139.00

Viewmaster 80

There used to be about a dozen 80 column cards for the Apple, now there's only ONE.

- TOTALLY Videx Compatible.
- 80 characters by 24 lines, with a sharp 7x9 dot matrix.
- On-board 40/80 soft video switch with manual 40 column override
- Fully compatible with ALL Apple languages and software—there are NO exceptions.
- Low power consumption through the use of CMOS devices.
- All connections are made with standard video connectors.
- Both upper and lower case characters are standard.
- All new design (using a new Microprocessor based C.R.T. controller) for a beautiful razor sharp display.
- The VIEWMASTER incorporates all the features of all other 80 column cards, plus many new improvements.

PRICE	BUSES IN	SWIFT AS	SWIFT AS SUPPORT	VIDEOPOWER	INVERSE CHARACTERS							
VIEWMASTER	179	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES
SUPERTRIS	MORE	NO	YES	NO	YES							
WIZARD80	MORE	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO
UNISON	MORE	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO
OMNIBUS	MORE	NO	YES	NO	YES							
VIEWMASTER	MORE	YES	YES	NO								
SUPERTRIS	MORE	YES	YES	NO								
VIDEOGRAPH	MORE	NO	NO	YES	NO	YES	NO	YES	NO	NO	NO	NO

The VIEWMASTER 80 works with all 80 column applications including CP/M, Pascal, WordStar, Format II, EasyWriter, Apple Writer II, VisiCalc, and all others. The VIEWMASTER 80 is THE MOST compatible 80 column card you can buy at ANY price!

PRICE \$179.00

MemoryMaster IIe 128K RAM Card

- Expands your Apple IIe to 192K memory.
- Provides an 80 column text display.
- Compatible with all Apple IIe 80 column and extended 80 column card software (same physical size as Apple's 64K card).
- Can be used as a solid state disk drive to make your programs run up to 20 times FASTER (the 64K configuration will act as half a drive).
- Permits your IIe to use the new double high resolution graphics.
- Automatically expands Visicalc to 95 K storage in 80 columns! The 64K config. is all that's needed. 128K can take you even higher.
- PRO-DOS will use the MemoryMaster IIe as a high speed disk drive.

- Precision software disk emulation for Basic, Pascal and CP/M is available at a very low cost. NOT copy protected.
- Documentation included, we show you how to use all 192K.

If you already have Apple's 64K card, just order the MEMORYMASTER IIe with 64K and use the 64K card to help to give you a full 128K. (The board is fully socketed so you simply plug in more chips.)

MemoryMaster IIe with 128K	\$249
Upgradable MemoryMaster IIe with 64K	\$169
Non-Upgradable MemoryMaster IIe with 64K	\$149

Our boards are far superior to most of the consumer electronics made today. ALL IC's are in high quality, socketed with mil spec. components used throughout. P.C. boards are glass epoxy with gold contacts. Made in America to be the best in the world. All products work in the APPLE II, II+, IIe, and Franklin. The MemoryMaster IIe is the only Applied Engineering also manufactures a full line of data acquisition and control products for the Apple A/D converters and digital I/O cards, etc. Please call for more information. All our products are fully tested with complete documentation and available for immediate delivery. All products are guaranteed with a no hassle THREE YEAR WARRANTY.

Texas Residents Add 5% Sales Tax
Add \$10.00 If Outside U.S.A.
Dealer Inquiries Welcome

Send Check or Money Order to:
APPLIED ENGINEERING
P.O. Box 798
Carrollton, TX 75006

Call (214) 492-2027
8 a.m. to 11 p.m. 7 days a week
MasterCard, Visa & C.O.D. Welcome
No extra charge for credit cards

9453- 20 72 95 2090 .1 JSR FIND.NEXT.STRING.VARIABLE
 9456- BO 44 2100 BCS .5 NO MORE VARIABLES
 9458- A0 00 2110 LDY #0 POINT AT LENGTH BYTE OF DESC.
 945A- B1 9B 2120 LDA (LOWTR),Y
 945C- FO F5 2130 BEQ .1 STRING LEN =0
 2140 *----CHECK LOCATION OF STRING----
 945E- AA 2150 TAX SAVE STRLEN IN X-REG
 945F- C8 2160 INY IF STRING DATA INSIDE PROGRAM,
 9460- B1 9B 2170 LDA (LOWTR),Y THEN NO NEED TO FIDDLE
 9462- 85 08 2180 STA FORPNT WITH IT FURTHER.
 9464- C5 69 2190 CMP VARTAB
 9466- C8 2200 INY
 9467- B1 9B 2210 LDA (LOWTR),Y
 9469- 85 09 2220 STA FORPNT+1
 946B- E5 6A 2230 SBC VARTAB+1 IN PROGRAM?
 946D- 90 E4 2240 BCC .1 ...YES, SO PASS
 2250 *----CHECK FOR SHORT STRING----
 946F- EO 03 2260 CPX #3 IF 1 OR 2, SPECIAL TREATMENT
 9471- BO 15 2270 BCS .3 ...LONG STRING
 2280 *----SHORT STRING HANDLER----
 9473- 86 06 2290 STX SHORT.FLAG NON-ZERO TO FLAG
 9475- A9 FF 2300 LDA #\$FF
 9477- 91 9B 2310 STA (LOWTR),Y MARKER IN 3RD DESC. BYTE
 9479- 88 2320 DEY POINT AT 2ND DESC. BYTE
 947A- CA 2330 DEX CHECK LENGTH
 947B- FO 02 2340 BEQ .2 LEN=1, PUT \$FF IN 2ND BYTE
 947D- B1 08 2350 LDA (FORPNT),Y LEN=2, SAVE CHAR IN 2ND BYTE
 947F- 91 9B 2360 .2 STA (LOWTR),Y
 9481- 88 2370 DEY POINT AT 1ST BYTE OF DESC.
 9482- B1 08 2380 LDA (FORPNT),Y MOVE FIRST BYTE OF STRING
 9484- 91 9B 2390 STA (LOWTR),Y TO DESC.
 9486- 10 CB 2400 BPL .1 ALWAYS
 2410 *----LONG STRING HANDLER----
 9488- B1 08 2420 .3 LDA (FORPNT),Y MARK FIRST BYTE OF STRING
 948A- 09 80 2430 ORA #\$80 MAKE NEG ASCII
 948C- 91 08 2440 .4 STA (FORPNT),Y
 948E- 88 2450 DEY BACK UP TOWARD BEG. OF DATA
 948F- 30 C2 2460 BMI .1 ..FINISHED MARKING THIS
 9491- B1 08 2470 LDA (FORPNT),Y SAVE STRING CHAR IN DESC.
 9493- C8 2480 INY
 9494- 91 9B 2490 STA (LOWTR),Y IN LAST 2 BYTES
 9496- 88 2500 DEY OF DESCRIPTOR
 9497- B9 9B 00 2510 LDA LOWTR,Y SAVE ADDR INSIDE STRING
 949A- BO FO 2520 BCS .4 ALWAYS SET
 2530 *----FINISHED MARKING STRINGS----
 949C- 60 2540 .5 RTS
 2550 *-----
 2560 * MOVE THE STRINGS AS HIGH AS POSSIBLE
 2570 *-----
 2580 RAISE.ALL.ACTIVE.STRINGS
 949D- 20 34 95 2590 JSR SET.STRING.POOL.STROLL
 94A0- 86 9C 2600 STA LOWTR+1 STARTS AT TOP
 94A2- 85 9B 2610 STA LOWTR OF STRNG POOL
 94A4- 20 3D 95 2620 .1 JSR FIND.NEXT.NEG.BYTE.IN.POOL
 2630 *-----
 2640 * CARRY CLEAR ON RETURN WHEN THRU
 2650 *-----
 94A7- 90 48 2660 BCC .4 ...NO MORE STRINGS IN POOL
 94A9- A0 00 2670 LDY #0
 94AB- 29 7F 2680 AND #\$7F
 94AD- 91 71 2690 STA (FRESPC),Y
 2700 *-----
 2710 * RESTORE STRING POOL TO POS ASC
 2720 * THEN RESET POINTERS
 2730 *-----
 94AF- 38 2740 SEC
 94B0- A5 71 2750 LDA FRESPC RECOVER ADDR.
 94B2- E9 02 2760 SBC #2 OF DESCRIPTOR
 94B4- 85 71 2770 STA FRESPC FROM THE STR
 94B6- B0 02 2780 BCS .2 ...NO BORROW
 94B8- C6 72 2790 DEC FRESPC+1
 94BA- B1 71 2800 .2 LDA (FRESPC),Y
 94BC- 85 08 2810 STA FORPNT AND PUT IT
 94BE- C8 2820 INY IN FORPNT
 94BF- B1 71 2830 LDA (FRESPC),Y
 94C1- 85 09 2840 STA FORPNT+1
 94C3- C8 2850 INY Y=2
 94C4- B1 08 2860 LDA (FORPNT),Y

```

2870 *-----*
2880 * RESTORE STRING BY RETURNING
2890 * THE FIRST TWO BYTES WHICH WERE
2900 * STORED IN THE DESCRIPTOR.
2910 *
2920 * THEN POINT DESCRIPTOR TO THE
2930 * NEW, CORRECT STRING POSITION.
2940 *-----*
94C6- 88 2950 DEY
94C7- 91 71 2960 STA {FRESPC},Y
94C9- B1 08 2970 LDA {FORPNT},Y
94CB- 88 2980 DEY Y=0
94CC- 91 71 2990 STA {FRESPC},Y
94CE- B1 08 3000 LDA {FORPNT},Y
94DO- 85 07 3010 STA STRING.LENGTH
94D2- 38 3020 SEC
94D3- A5 9B 3030 LDA LOWTR
3040 *-----*
3050 * POINT LOWTR & STRING DESCRIPTOR
3060 * TO BOTTOM OF NEW STRING POOL.
3070 *
3080 * LOWTR HOLDS THE MOVING BOTTOM
3090 * OF THE STRING POOL.
3100 *-----*
94D5- B5 07 3110 SBC STRING.LENGTH
94D7- 85 9B 3120 STA LOWTR
94D9- C8 3130 INY
94DA- 91 08 3140 STA {FORPNT},Y
94DC- A5 9C 3150 LDA LOWTR+1
94DE- B9 00 3160 SBC #0
94EO- 85 9C 3170 STA LOWTR+1
94E2- C8 3180 INY
94E3- 91 08 3190 STA {FORPNT},Y
3200 *-----*
3210 * NOW MOVE THE STRING TO ITS
3220 * NEW ADDRESS.
3230 *-----*
94E5- A4 07 3240 LDY STRING.LENGTH
94E7- 88 3250 DEY
94E8- B1 71 3260 .3 STA {FRESPC},Y
94EA- 91 9B 3270 STA {LOWTR},Y
94EC- 88 3280 DEY
94ED- 10 F9 3290 BPL .3
94EF- 30 B3 3300 BMI .1 ...ALWAYS
3310 *----FINISHED MOVING STRINGS----*
94F1- 60 3320 .4 RTS
3330 *-----*
3340 * RESTORE NORMAL CONFIGURATION OF PNTR AND DATA
3350 * FOR STRINGS OF 1 OR 2 CHARACTERS
3360 *
3370 *-----*
3380 * SCAN THRU VARIABLE SPACE AGAIN:
3390 * DESCRIPTORS OF STRINGS MARKED WITH $FF
3400 * CONTAIN THE CHAR(S) TO RESTORE TO POOL.
3410 *-----*
3420 * FRESPC POINTS AT BOTTOM OF POOL
3430 * LOWTR POINTS AT DESCRIPTORS
3440 *-----*
94F2- 20 65 95 3450 FIX.SHORT.STRINGS
94F5- 20 72 95 3460 .1 JSR INITIATE.SEARCH
94F8- B0 39 3470 BCS .5 ...FINISHED!
94FA- A0 02 3480 LDY #2 ...POINT AT 3RD BYTE, 2ND OF ADDR
94FC- 84 07 3490 STY STRING.LENGTH
94FE- B1 9B 3500 LDA (LOWTR),Y IF 3RD BYTE =$FF, SHORTY.
9500- C9 FF 3510 CMP #$FF A SHORTY?
9502- D0 F1 3520 BNE .1 ...NO, KEEP SCANNING VARIABLES
9504- 88 3530 DEY ...YES, POINT AT 2ND BYTE
9505- B1 9B 3540 LDA (LOWTR),Y IF 2ND BYTE ALSO $FF,
9507- 48 3550 PHA THEN LEN=1
9508- 10 02 3560 BPL .2 ...NOT $FF, ITS A STR CHAR
950A- C6 07 3570 DEC STRING.LENGTH
950C- 88 3580 .2 DEY ...POINT AT 1ST BYTE OF DESCRIPTOR
950D- B1 9B 3590 LDA (LOWTR),Y GET 1ST ASC CHAR OF STRING
950F- 48 3600 PHA SAVE ON STACK
3610 *----CALC PLACE IN POOL FOR DATA--*
9510- 38 3620 SEC
9511- A5 71 3630 LDA FRESPC REPOINT FRESPC
9513- E5 07 3640 SBC STRING.LENGTH
9515- 85 71 3650 STA FRESPC
9517- B0 02 3660 BCS .3
9519- C6 72 3670 DEC FRESPC+1

```

3680 *---RESTORE LENGTH TO DESC.----
 3690 .3 LDA STRING.LENGTH
 3700 STA (LOWTR),Y
 3710 *---STORE CHARS INTO POOL-----
 3720 *---AND ADDR INTO DESCRIPTOR-----
 3730 PLA FIRST CHAR
 3740 STA (FRESPC),Y
 3750 INY
 3760 LDA FRESPC LOBYTE OF ADDR
 3770 STA (LOWTR),Y
 3780 PLA 2ND CHAR
 3790 BMI .4 ...IT IS \$FF, ONLY 1 CHAR
 3800 STA (FRESPC),Y
 3810 .4 INY
 3820 LDA FRESPC+1 HIBYTE OF ADDR
 3830 STA (LOWTR),Y
 3840 BNE .1 ALWAYS
 3850 *---ALL FINISHED WITH SHORTIES---
 3860 .5 RTS
 3870 *-----
 3880 * STRING POOL STROLL
 3890 *-----
 3900 SET.STRING.POOL.STROLL
 3910 LDX MEMSIZE+1 POINT FRESPC
 3920 LDA MEMSIZE AT HIMEM
 3930 STA FRESPC TO START
 3940 STX FRESPC+1 STROLL.
 3950 RTS
 3960 *-----
 3970 * SEARCH STRING POOL FROM TOP TO BOTTOM
 FOR A NEGATIVE BYTE.
 3980 *-----
 4000 * RETURN .CS. IF NEG BYTE FOUND,
 4010 * .CC. IF REACHED END OF POOL
 4020 *-----
 4030 FIND.NEXT.NEG.BYTE.IN.POOL
 4040 LDX FRESPC+1
 4050 LDY FRESPC
 4060 LDA #0 PAGE AT A TIME
 4070 STA FRESPC
 4080 TYA IS IT ZERO?
 4090 BNE .2 NO!
 4100 .1 DEX YES
 4110 CPX FRETOP+1 STILL IN POOL?
 4120 BCC .5 ..NO
 4130 STX FRESPC+1 DO NEXT PAGE
 4140 .2 DEY
 4150 BEQ .3
 4160 LDA (FRESPC),Y
 4170 BPL .2 POS ASCII
 4180 BMI .4 NEG SO QUIT
 4190 .3 LDA (FRESPC),Y
 4200 BPL .1 NEW PAGE
 4210 .4 CPX FRETOP+1
 4220 BNE .5 FRESPC>FRETOP
 4230 CPY FRETOP FOR CARRY FLAG
 4240 .5 STY FRESPC FRESPC POINTS TO NEG ASC
 4250 RTS
 4260 *-----
 4270 * SET UP SEARCH OF VAR TABLE
 4280 *-----
 4290 INITIATE.SEARCH
 4300 LDA VARTAB START AT BEGINNING OF VARIABLES
 4310 STA INDEX
 4320 LDX VARTAB+1
 4330 STX INDEX+1
 4340 LDY #7 EACH VAR TAKES 7 BYTES
 4350 STY OFFSET
 4360 RTS
 4370 *-----
 4380 * FIND NEXT STRING VARIABLE
 4390 *-----
 4400 FIND.NEXT.STRING.VARIABLE
 4410 .1 LDX INDEX+1 SETUP SEARCH FOR NEXT STRING
 4420 LDA INDEX
 4430 LDY OFFSET
 4440 CPY #7 STILL IN SIMPLE VARIABLES?

QUICKTRACE

relocatable program traces and displays the actual machine operations, while it is running without interfering with those operations. Look at these FEATURES:

Single-Step mode displays the last instruction, next instruction, registers, flags, stack contents, and six user-definable memory locations.

Trace mode gives a running display of the Single-Step information and can be made to stop upon encountering any of nine user-definable conditions.

Background mode permits tracing with no display until it is desired. Debugged routines run at near normal speed until one of the stopping conditions is met, which causes the program to return to Single-Step.

QUICKTRACE allows changes to the stack, registers, stopping conditions, addresses to be displayed, and output destinations for all this information. All this can be done in Single-Step mode while running.

Two optional display formats can show a sequence of operations at once. Usually, the information is given in four lines at the bottom of the screen.

QUICKTRACE is completely transparent to the program being traced. It will not interfere with the stack, program, or I/O.

QUICKTRACE is relocatable to any free part of memory. Its output can be sent to any slot or to the screen.

QUICKTRACE is completely compatible with programs using Applesoft and Integer BASICs, graphics, and DOS. (Time dependent DOS operations can be bypassed.) It will display the graphics on the screen while QUICKTRACE is alive.

QUICKTRACE is a beautiful way to show the incredibly complex sequence of operations that a computer goes through in executing a program

*See these programs at participating Computerland and other
fine computer stores.*

**Anthro - Digital Software, Inc.
P.O. Box 1385 Pittsfield, MA 01202**

QUICKTRACE

\$50

Is a trademark of Anthro-Digital, Inc.

Copyright © 1981

Written by John Rogers

957A-	D0	18	4450	BNE .4	...NO, IN ARRAYS
957C-	E4	6C	4460	CPX ARYTAB+1	WE WERE, CHECK FURTHER...
957E-	90	04	4470	BCC .2	...YES, STILL SIMPLE
9580-	C5	6B	4480	CMP ARYTAB	
9582-	BO	0A	4490	BCS .3	...NO
9584-	20	E6	95	4500	.2 JSR IS.THIS.A.STRING.VARIABLE
9587-	BO	4C	4510	BCS .8	...STRING FOUND
9589-	20	D9	95	4520	JSR NXTEL NOT A STRING
958C-	90	E4	4530	BCC .1	...ALWAYS, TRY AGAIN
958E-	46	1B	4540	LSR OFFSET	SET OFFSET TO 3 NOW
9590-	85	1D	4550	STA ARRAY.END	
9592-	86	1E	4560	STX ARRAY.END+1	
9594-	E4	1E	4570	CPX ARRAY.END+1	INSIDE AN ARRAY?
9596-	90	3D	4580	BCC .8	...YES
9598-	C5	1D	4590	CMP ARRAY.END	
959A-	90	39	4600	BCC .8	
959C-	E4	6E	4610	CPX STREND+1	STILL IN VAR SPC?
959E-	90	05	4620	BCC .5	...YES
95A0-	C5	6D	4630	CMP STREND	
95A2-	90	01	4640	BCC .5	...YES
95A4-	60		4650	RTS	CARRY SET WHEN THRU VAR SPC
			4660	*---SET UP A NEW ARRAY-----	
95A5-	A0	02	4670	.5 LDY #2	
95A7-	18		4680	CLC	
95A8-	B1	19	4690	LDA (INDEX),Y	
95AA-	65	19	4700	ADC INDEX	
95AC-	85	1D	4710	STA ARRAY.END	POINTER TO
95AE-	C8		4720	INY	NEXT ARRAY
95AF-	B1	19	4730	LDA (INDEX),Y	
95B1-	65	1A	4740	ADC INDEX+1	
95B3-	85	1E	4750	STA ARRAY.END+1	
95B5-	20	E6	95	4760	JSR IS.THIS.A.STRING.VARIABLE IS THIS A STR?
95B8-	BO	0A	4770	BCS .6	...YES
95BA-	A5	1D	4780	LDA ARRAY.END	
95B1-	85	19	4790	STA INDEX	NO
95BE-	A6	1E	4800	LDX ARRAY.END+1	
95C0-	86	1A	4810	STX INDEX+1	
95C2-	DO	DO	4820	BNE .4	...ALWAYS
			4830	*---FOUND STRING ARRAY-----	
95C4-	A0	04	4840	.6 LDY #4	POINT AT
95C6-	B1	19	4850	LDA (INDEX),Y	#DIMENSIONS OF ARRAY
95C8-	OA		4860	ASL #2	
95C9-	69	05	4870	ADC #5	
95CB-	65	19	4880	ADC INDEX	POINT INDEX TO
95CD-	85	19	4890	STA INDEX	FIRST ELEMENT
95CF-	90	02	4900	BCC .7	OF NEW ARRAY
95D1-	E6	1A	4910	INC INDEX+1	
95D3-	A6	1A	4920	LDX INDEX+1	
			4930	*---NEXT VARIABLE-----	
95D5-	85	9B	4940	.8 STA LOWTR	LOWTR->STR DESCRIPTOR
95D7-	86	9C	4950	STX LOWTR+1	
			4960	*---NEXT VARIABLE-----	
95D9-	18		4970	NXTTEL CLC	
95DA-	A5	1B	4980	LDA OFFSET	POINT INDEX TO
95DC-	65	19	4990	ADC INDEX	NEXT VAR OR ELEMENT
95DE-	85	19	5000	STA INDEX	
95E0-	90	03	5010	BCC .1	
95E2-	E6	1A	5020	INC INDEX+1	
95E4-	18		5030	CLC	
95E5-	60		5040	.1 RTS	STR FOUND,CARRY CLEAR
			5050	*---SUBROUTINE STRING CHECK-----	
			5060	IS.THIS.A.STRING.VARIABLE	
			5070	LDY #0	
95E6-	A0	00	5090	CLC	INCASE NOT STR
95E8-	18		5100	LDA (INDEX),Y	
95E9-	B1	19	5110	BMI .2	...NOT STRING
95EB-	30	0D	5120	INY	
95ED-	C8		5130	LDA (INDEX),Y	
95EE-	B1	19	5140	BPL .2	...NOT STRING
95F0-	10	08	5150	LDA #2	POINT PAST STR NAME
95F2-	A9	02	5160	ADC INDEX	
95F4-	65	19	5170	BCC .1	STRING
95F6-	90	01	5180	INX INDEX+1	
95F8-	E8		5190	SEC	CARRY SET IF STR FOUND
95F9-	38		5200	.1 RTS	
95FA-	60		5210	.2	
			5220	-----	

Changing VERIFY to DISPLAY.....Bob Sander-Cederlof

In the July 1982 issue of AAL we showed how to make a text file display command inside DOS. Bob Bragner added 80-column output to it in the July 1983 issue. The Dec 1983 InCider printed an article by William G. Wright about a DOS modification that provided text file display without removing any previous features.

Wright's patches modify the VERIFY command so that as sectors are being verified, if the file is a text file, they are displayed on the screen or printer. If there are any \$00 bytes in a sector, they are merely skipped over, so his patches will handle some random access files, as well as sequential. Non-text files are still verified in the normal manner.

I was prompted by his article to write up another little program. This one will hook into the VERIFY processor in the file manager when you BRUN the program. Later, 30BG from the monitor or CALL 779 from Applesoft will dis-install the patch. My patch modifies VERIFY so that as each sector of a file is verified it is displayed in hexadecimal on the screen or a printer. I do not distinguish between text and non-text files, although it would be a simple matter to do so. As with Wright's patches, random access files may also be displayed, up to the first hole in the track/sector list.

The creative among you will want to add many bells and whistles to my little program. Perhaps 80-column display, showing an entire sector at a time rather than half a sector. Perhaps display of the bytes in both hex and ASCII on the same line. Perhaps the ability to scan back and forth through a file, using the arrow keys. All these are possible, and not too difficult. Have fun!

```
1000 *SAVE S.DISPLAY FILE
1010 *
1020 *----- PATCH DOS TO CHANGE VERIFY
1030 *----- INTO DISPLAY
1040 *
F941- 1050 MON.PRNTAX .EQ $F941
F94A- 1060 MON.PRL2 .EQ $F94A
FD6E- 1070 MON.CROUT .EQ $FD8E
FDDA- 1080 MON.PRBYTE .EQ $FDDA
FDED- 1090 MON.COUT .EQ $FDED
1100 *
1110 .OR $300
1120 *
1130 PATCH LDA #DISPLAY HOOK INTO VERIFY COMMAND
0300- A9 16 1140 STA $AD1C
0302- BD 1C AD 1150 LDA /DISPLAY
0305- A9 03 1160 STA $AD1D
0307- BD 1D AD 1170 RTS
030A- 60 1180 *
1190 UNPATCH
030B- A9 B6 1200 LDA #$BOB6 RESTORE NORMAL VERIFY
030D- BD 1C AD 1210 STA $AD1C
0310- A9 B0 1220 LDA /$BOB6
0312- BD 1D AD 1230 STA $AD1D
0315- 60 1240 RTS
1250 *
1260 DISPLAY
0316- 20 8E FD 1270 JSR MON.CROUT START SECTOR WITH <RET>
0319- 20 B6 B0 1280 JSR $BOB6 READ NEXT SECTOR
031C- B0 09 1290 BCS .1 END OF FILE
031E- A0 00 1300 LDY #0 DISPLAY FIRST HALF SECTOR
```

0320-	20	28	03	1310	JSR SHOW	
0323-	20	28	03	1320	JSR SHOW	DISPLAY SECOND HALF
0326-	18			1330	CLC	SIGNAL NOT END OF FILE
0327-	60			1340	RTS	
				1350	*	
0328-	AD	E5	B5	1360	SHOW	LDA \$B5E5 DISPLAY SECTOR POSITION
0329-	AE	E4	B5	1370		LDX \$B5E4
032E-	20	41	F9	1380		JSR MON.PRNTAX
0331-	A9	10		1390		LDA #16 16 LINES PER BLOCK
0333-	8D	65	03	1400		STA LCNT
0336-	D0	05		1410		BNE .2
0338-	A2	04		1420	.1	LDX #4 PRINT 4 BLANKS
033A-	20	4A	F9	1430		JSR MON.PRL2 SO COLUMNS LOOK NEATER
033D-	A9	08		1440	.2	LDA #8 8 BYTES PER LINE
033F-	8D	64	03	1450		STA BCNT
0342-	98			1460		TYA PRINT BYTE COUNT
0343-	20	DA	FD	1470		JSR MON.PRBYTE
0346-	A9	AD		1480		LDA "#--" PRINT "--"
0348-	20	ED	FD	1490		JSR MON.COUT
034B-	A9	A0		1500	.3	LDA "# " PRINT " "
034D-	20	ED	FD	1510		JSR MON.COUT
0350-	B1	42		1520		LDA (\$42),Y NEXT BYTE FROM FILE
0352-	C8			1530		INY
0353-	20	DA	FD	1540		JSR MON.PRBYTE
0356-	CE	64	03	1550		DEC BCNT
0359-	D0	F0		1560		BNE .3 MORE TO THIS LINE
035B-	20	8E	FD	1570		JSR MON.CROUT NEXT LINE
035E-	CE	65	03	1580		DEC LCNT
0361-	D0	D5		1590		BNE .1
0363-	60			1600		RTS
				1610	*	
0364-				1620	BCNT :BS 1	
0365-				1630	LCNT :BS 1	
				1640	*	

Complete, Commented Source Code!

Our software is not only unlocked and fully copyable
...we often provide the complete source code on disk, at unbelievable prices!

S-C Macro Assembler. The key to unlocking all the mysteries of machine language. Combined editor/assembler with 29 commands, 20 directives. Macros, conditional assembly, global replace, edit, and more. Highest rating "The Book of Apple Software" in 1983 and 1984. \$80.

Powerful cross-assembler modules also available to owners of S-C Macro Assembler. You can develop software on your Apple for 6800, 6805, 6809, 68000, 8085, 8048, 8051, 1802, LSI-11, and Z-80 microprocessors. \$50 each.

S-C Xref. A support program which works with the S-C Macro Assembler to generate an alphabetized listing of all labels in a source file, showing with each label the line number where it is defined along with all line numbers containing references to the label. You get the complete source code for this amazingly fast program, on disk in format for S-C Macro Assembler. \$50.

Full Screen Editor. Integrates with the built-in line-oriented editor in the S-C Macro Assembler to provide a powerful full-screen editor for your assembly language source files. Drivers for Videc, STB80, and Apple IIe 80-column boards are included, as well as standard 40-column version. Requires 64K RAM in your Apple. Complete source code on disk included. \$50.

S-C Docu-Mentor for Applesoft. Complete documentation of Applesoft internals. Using your ROM Applesoft, produces ready-to-assemble source code with full labels and comments. Educational, entertaining, and extremely helpful. Requires S-C Macro Assembler and two disk drives. \$50.

S-C Word Processor. The one we use for manuals, letters, our monthly newsletter, and whatever. 40-columns only, requires lower-case display and shiftkey mod. Works with standard DOS text files, but at super fast (100 sectors in 7 seconds). No competition to WordStar, but you get complete source code! \$50.

Apple Assembly Line. Monthly newsletter published since October, 1980, for assembly language programmers or those who would like to be. Tutorial articles, advanced techniques, handy utility programs, and commented listings of code in DOS, ProDOS, and the Apple ROMs. Helps you get the most out of your Apple! \$18/year.

S-C SOFTWARE CORPORATION
2331 Gus Thomasson, Suite 125
Dallas, TX 75228 (214) 324-2050

Professional Apple Software Since 1978
Visa, MasterCard, American Express, COD accepted.
Apple is a trademark of Apple Computer Inc.



Changing Tab Stops in the 68000 Cross Assembler.....
Bob Sander-Cederlof

The procedure as described in the S-C Macro Assembler manual works for the 6502 version and for all the cross assemblers except the 68000 cross assembler. The procedure described in Appendix D will not work because the 68000 cross assembler uses both banks of memory at \$D000-DFFF. In order to be certain the correct one is switched on, the command interpreter keeps using the selection soft switches. The result is that the bank stays write-protected, and no patches ever get installed.

Of course, there is a simple way around the problem. Here is how to change the tab stops in the 68000 Cross Assembler:

First, boot the cross assembler disk and select option 2, loading the language card version at \$D000.

```
:BLOAD S-C.ASM.MACRO.68000.LC
:MNTR
*AA60.AA61
*AA60- xx yy  (probably C6 27)
*D010.D014
D010- OE 16 1B 20 00
*C083 C083 D010:7 10 1B 2B  (or whatever values you like)
C083- zz
C083- zz
*D010.D014
D010- 07 10 1B 2B 00
*BSAVE S-C.ASM.MACRO.68000.LC,A$D000,L$yyxx
*3DOG
:      that's it!
```

Similar methods apply to the other customizing patches mentioned in Appendix D.

OBJ.APWRT] [F

Detailed, complete, and thorough disassembly script includes full details on customizing Applewriter IIe and capturing your own source code for modification.

TWO unlocked, jammed-full and double-sided diskettes plus a free "must have" bonus book for only \$29.95.

Now six sides
SOURCECODE
8 Synergetics
Box 1300-AAL
Thatcher, AZ 85552
(602) 428-4073

VISA and MASTERCHARGE accepted.

S-C Macro and GPLE.LC on the //e.....Bob Bragner
Istanbul, Turkey

I've long been bothered by the way loading the S-C Macro Assembler wipes out GPLE (Neil Konzen's Global Program Line Editor) when you go from Applesoft to Assembler. It shouldn't happen, because GPLE resides in the alternate bank at \$D000, not used by Macro. I've also been unhappy that the //e version of S-C Macro Assembler doesn't have the automatic line of dashes provided by <esc> L after a line number when you are in 80-column mode.

Just the other day I discovered by accident that all is not lost. If things are done in just the right sequence both of my peeves vanish.

First load up the S-C Macro Assembler into the \$D000 area. Then enter Applesoft by typing the FP command, and BRUN GPLE.LC. Initialize the 80-column card with ctrl-V and enter the assembler by typing the INT command. This leaves GPLE connected so that the assembler sees the <esc> L command. Try it by typing a line number and <esc> L.

It also allows the assembler to see <esc> L to turn a catalog line into a LOAD command, but due to the way the word LOAD is poked onto the screen you get L O A D which clobbers the file name. (I never use the automatic load anyway, so this does not bother me.)

RESET will partially disable GPLE.LC, but you can restore it by typing the & command from Applesoft. If you want RESET to NOT molest GPLE, change the reset vector to \$B6B3. You can do this from the monitor with "3F2:B3 B6 13", or from S-C Macro with "RST \$B6B3".

I don't know why this all works, but I think it has something to do with the way the 80-column card initializes itself by copying the //e's monitor ROM into the \$F800-FFFF space of RAM.

By the way, GPLE uses some patch space inside DOS 3.3 which is also used by the fast DOS text file I/O patch, so beware of mixing them.

Will Rockwell 65C02's work in an old Apple.....Bob Stout

Not unless you have the 2 MHz part. For some reason the timing is too tight and slightly different to use a 1MHz 65C02, unless you have an Apple //e. The 2 MHz chips WILL work in Apple II and II Plus.

Redundancy in Tables for Faster Lookups....Bob Sander-Cederlof

When speed is the main objective, you can sometimes use table lookups to great advantage. You trade program size for speed.

Here is an easy example. Suppose I want to convert the two nybbles of a byte to ASCII characters. I can do it all with code, like this:

CONVERT

PHA	Save original byte
LSR	Position first nybble
LSR	
LSR	
JSR MAKE.ASCII	
STA XXX	
PLA	Original byte
AND #\$0F	Isolate second nybble
JSR MAKE.ASCII	
STA XXX+1	
RTS	

MAKE.ASCII

ORA #\$B0	Make B0...BF
CMP #\$BA	
BCC .1	It is 0-9
ADC #6	Make A-F codes
.1 RTS	

That takes 30 bytes, and 75-77 cycles including a JSR CONVERT to call it. Actually 75 cycles if both nybbles are 0-9, 77 cycles if they both are A-F, and 76 cycles if there is one of each. If I move the code from MAKE.ASCII in-line, it saves 24 cycles (two JSRs, two RTSs), and only lengthens the program by one byte.

Or I can do a table lookup by substituting these two lines for both JSR MAKE.ASCII lines above:

```
TAX  
LDA ASCII.TABLE,X
```

and making a little table like this:

```
ASCII.TABLE .AS /0123456789ABCDEF/
```

In this form, the program takes 49 cycles, and uses a total of 39 bytes including the table. Perhaps it could be an advantage that the # of cycles is always constant, regardless of the value being converted.

You can make it even faster by using two whole pages for table space, like this:

CONVERT

```
TAX  
LDA HI.TABLE,X
```

```

STA XXX
LDA LO.TABLE,X
STA XXX+1
RTS

HI.TABLE
.AS -/0000000000000000/
.AS -/1111111111111111/
.
.
.AS -/FFFFFFFFFFFFFFFFFF/

LO.TABLE
.AS -/0123456789ABCDEF/
.AS -/0123456789ABCDEF/
.
.
.AS -/0123456789ABCDEF/

```

The program itself is 14 bytes long, but there are 512 bytes of tables. The conversion, including JSR and RTS, now takes only 30 cycles. And since the program is now so short, it would probably get placed in line, saving the JSR-RTS, converting in only 18 cycles. And if the in-line routine already had the nybble in the X-reg, whack off another two cycles.

The redundancy in the tables gives a huge speed increase.

I have been tearing into the super fast copy utility that comes with Locksmith 5.0, and I discovered some of these redundancy tricks in their disk I/O tables. For example, the table for converting a six-bit value into a disk-code normally takes 64 bytes. The table looks like this:

```

TABLE .HS 96979A9B9D9E9FA6
.
.
.HS F7F9FAFBFCFDFFEFF

```

Code to access the table might look like this:

```

LDA BUFFER,X
AND #$3F      Mask to 6 bits
TAY
LDA TABLE,Y

```

When you are writing to a disk, every single cycle counts. Therefore, it is pleasant to discover redundant tables. By making four copies of the table, using 256 bytes rather than 64, we no longer need to strip off the first two bits. The code can be shortened to this:

```

LDY BUFFER,X
LDA TABLE,Y

```

It only saves 3 cycles, but those three cycles can and do make the whole difference in the fast copy program. That is part of Locksmith's secret to reading a whole disk into RAM in only 8 seconds.

Speaking of Locksmith.....Warren R. Johnson

Did you know that Locksmith 5.0 can nearly be copied by plain old COPYA? Or with its own fast backup copier? All but the last few tracks copy, and they may not be necessary.

The only problem is, the resulting copy will not boot until you make a small patch using some sort of disk ZAP utility. (You can use Omega's Inspector/Watson team, Bag of Tricks, Disk Fixer, CIA, for example.) Patch Track-0F Sector-0E Byte-6F: change it from 6C to 0F. [Editor's note: in my copy, Locksmith had C6 in that byte rather than 6C. And I have not tried the resulting disk to see if all functions work.]

I have modified my Apple a little to make my life easier. I have 2732's in the motherboard ROM sockets, with bank switch selection. Applesoft is in one bank, and a modified version of Applesoft in the other. My modifications include replacing the old cassette commands (LOAD/SAVE/SHLOAD etc.) for an INWAT command. INWAT downloads the Inspector and Watson from some expansion chassis ROM boards.

Lancaster's OBJ.APWRT][F.....Bob Sander-Cederlof

You may have noticed a little ad in the last few issues for an obscure title, "OBJ.APWRT][F". Don Lancaster, author of such favorites as Enhancing the Apple, Incredible Secret Money Machine, Micro Cookbook, etc., has torn into Applewriter //e. After a thorough analysis, he completely documented it, in the style of Beneath Apple DOS. The results, or at least part of them, will be chapter 12 in volume 2 of Enhancing.

He sent me a pre-print to look at and make comments about. My main comment is WOW! It doesn't matter if you like Applewriter or not. It doesn't even matter if you have never seen Applewriter. You still can learn a tremendous amount by reading through Don's text and comments. Of course it is better if you DO have Applewriter //e, because he tells you how to make some great customizing modifications.

You can get it all on disk for only \$29.95. Actually, it is not on "disk" ... it is on SIX disk sides, jam-packed full. Don even throws in a free book for good measure.

You can order OBJ.APWRT][F directly from Synergetics, or from S-C Software.

About Disk Drive Pressure Pads.....Bob Sander-Cederlof

After you have used your disk drive for six months or so, it will probably develop a scary noise or two. I know mine have.

My oldest drive is serial 1901 (the Shugart mechanics inside the box have a number somewhere in the low 400's). Every once and a while it will make the most dangerous sounding noise you ever heard, something like dragging rusty chains across the road. I have read in various magazines and newsletters that these noises are almost always caused by a dirty pressure pad.

The pressure pad rides on the top surface of the disk, pressing the disk surface down against the recording head. It is a 1/8 inch circle of felt glued to a slightly larger plastic stud. The shaft of the stud is split and tapered, so it will fit through a hole and lock in place. You can easily remove the pad and stud by pressing on the split end.

But where do you get new ones? Maybe at a computer store, but they sure don't keep them on display. I decided to try a little home maintenance, and it worked. I gently scraped the felt surface with the blade of my pocket knife, and all the old caked oxide turned to powder and fell off. Then I rubbed the oxide on a piece of paper, to smooth out the felt. After putting the drive all back together, it ran quietly.

It worked so well, I performed the operation on two more drives. And surprisingly, one drive which had been giving lots of errors was working accurately again.

A few other disk maintenance tips:

One particularly noisy drive a few years ago had loose screws trying to hold the drive motor down. A few wrist twists and all was well.

If a drive can read, but writes garbage, it is probably the 74LS125 on the analog board inside the drive. Replace that chip for 25 cents or so, and you have saved \$60 in repair bills.

Will ProDOS Work on a Franklin?.....Bob Stout

If you try to boot up ProDOS on a Franklin, it probably will fail. The ProDOS boot routine checks to see if you are in a genuine Apple monitor ROM. However, you can make it work.

Start the boot procedure; when meaningful action appears to have ceased, press the RESET switch. Get into the monitor and type 2647:EA EA and 2000G. Voila!

Rod's Color Pattern in 6502 Code.....Charles H. Putney

When I read the January AAL with Bob Urschel's article about running Rod's Color Pattern on the QWERTY 68000 board, it sounded like a challenge. You may remember I like speed challenges, at least inside computers.

Fifty times faster than Basic didn't sound too fast, so I checked a simple loop to see if it might be possible to save the dignity of the 6502. It did look possible, at least by using tricky table-driven code.

So, I wrote some more code and it looked like 8.0 seconds per loop. This clocks out at 55 times faster than Integer BASIC, but I didn't have the internal calculation for the color value exactly like the original.

I finally decided to use a table lookup for the color calculation. Now the problem was how to create all those data statements. I thought about using some macros, but the calculations are too involved. I wrote an Applesoft program to generate the lines of code for the assembler, and then EXECed them into my source code. I finally got all the bugs out and timed it.

The table-driven version performs a main loop every 6.2 seconds, compared to 446 seconds per loop for the Integer BASIC version. That is nearly 72 times faster.

Well, my only worry now is that Bob Urschel made an error in his timing, and his really runs 200 times faster. If not, we have saved face for the venerable 6502.

Of course, we did use a little more memory. But that is frequently a trade-off worth making in important programs.

For comparison purposes, here is the Integer BASIC program again:

```
10 GR
20 FOR W = 3 TO 50
30 FOR I = 1 TO 19
40 FOR J = 0 TO 19
50 K = I+J
60 COLOR = J*3 / (I+3) + I*W/12
70 PLOT I,K: PLOT K,I: PLOT 40-I,40-K:PLOT40-K,40-I
80 PLOT K,40-I: PLOT 40-I,K: PLOT I,40-K: PLOT 40-K,I
90 NEXT J: NEXT I: NEXT 2
100 GO TO 20
```

My program to generate the data tables includes similar logic. I broke the tables into two planes, rather than storing one data table $48*19*20 = 18,240$ bytes long. One plane computes $J*3/(I+3)$, and includes 380 bytes. The other computes $I*W/12$ and includes $48*19=912$ bytes. My table lookup routine uses I and J to index into the first plane, and I and W into the second. Then the two values are added together. Pretty tricky!

```

100 REM BUILD TABLES FOR ROD'S COLOR
110 D$ = CHR$(4)
120 PRINT D$"OPEN TTT": PRINT D$"WRITE TTT"
130 GOSUB 200
140 PRINT D$"CLOSE"
150 END
200 FOR I = 1 TO 19
210 PRINT "WI" LEFT$(STR$(I) + " ", 2) ".HS ";
220 FOR W = 3 TO 26: GOSUB 500: NEXT : PRINT
230 PRINT " " .HS ";
240 FOR W = 27 TO 50: GOSUB 500: NEXT : PRINT
250 NEXT I
300 FOR I = 1 TO 19
310 PRINT "JI" LEFT$(STR$(I) + " ", 2) ".HS ";
320 FOR J = 0 TO 19
330 C = INT(J * 3 / (I + 3)): C = C - INT(C / 16) * 16: IF C > 9 THEN
C = C + 7
340 PRINT "0" CHR$(C + 48);
350 NEXT : PRINT
360 NEXT I
370 RETURN
500 C = INT(I * W / 12): C = C - INT(C / 16) * 16: IF C > 9 THEN C = C
+ 7
510 PRINT "0" CHR$(C + 48);
520 RETURN

```

I believe in letting computers work for me, so I had to use some macros to simplify typing in all the code for those eight plot statements. I wrote a PLOT macro, but then I noticed that there was some redundant code that way. By rearranging the order of the PLOT statements, I can separate the y-setup from the x-setup and plot. That way the base address does not get re-calculated as often, saving more time. Here is my program:

```

1010 *SAVE S.PUTNEY'S COLOR
1020 .OR $6000
1030 .TF B.PUTNEY
1040 -----
1050 *
1060 *      FAST ROD'S COLOR PATTERN
1070 *
1080 *      CHARLES H. PUTNEY
1090 *      18 QUINNS ROAD
1100 *      SHANKILL
1110 *      CO. DUBLIN
1120 *      IRELAND
1130 *
1140 -----
1150 *
1160 *      PAGE ZERO ADDRESSES
1170 *
1180 INVI    .EQ $EE      VARIABLE 40 - I
1190 INVK    .EQ $EF      VARIABLE 40 - K
1200 POINTR  .EQ $F9      LORES PAGE POINTER (TWO BYTES)
1210 I       .EQ $FB      VARIABLE I
1220 J       .EQ $FC      VARIABLE J
1230 K       .EQ $FD      VARIABLE K
1240 W       .EQ $FE      VARIABLE W
0007-     1250 COLOR1 .EQ $07      HALF OF COLOR FORMULA
1260 -----
0008-     1270 COLOR   .EQ $08,09
0008-     1280 COLEVN  .EQ $08      EVEN ROW COLOR
0009-     1290 COLODD  .EQ $09      ODD ROW COLOR
000A-     1300 MASK    .EQ $0A,0B
000A-     1310 MSKODD  .EQ $0A
000B-     1320 MSKEVN  .EQ $0B
1330 *

```

----- APPLE SOFTWARE -----**NEW!!! FONT DOWNLOADER & EDITOR (\$39.00)**

Turn your printer into a custom typesetter. Downloaded characters remain active while printer is powered. Can be used with every word processor capable of sending ESC and control codes to the printer. Switch back and forth easily between standard and custom fonts. All special printer functions (like expanded, compressed, emphasized, underlined, etc.) apply to custom fonts. Full HIRES screen editor lets you create your own custom characters and special graphics symbols. Compatible with many 'dumb' & 'smart' printer I/F cards. User driver option provided. Specify printer: Apple Dot Matrix Printer, C.Itoh 8510A (Prowriter), Epson FX-80/100 or OkiData 92/93.

DISASM 2.2e - AN INTELLIGENT DISASSEMBLER (\$30.00)

Investigate the inner workings of machine language programs. DISASM converts 6502 machine code into meaningful, symbolic source. Creates a standard DOS 3.3 text file which is directly compatible with DOS ToolKit, LISA and S-C (4.0 and MACRO) assemblers. Handles data tables, displaced object code & even lets you substitute your own meaningful labels. (100 commonly used Monitor & Pg Zero pg names included.) An address-based cross reference table provides further insight into the inner workings of machine language programs. DISASM is an invaluable machine language learning aid to both the novice & expert alike. **SOURCE code: \$60.00**

S-C ASSEMBLER (Ver4.0 only) SUPPORT UTILITY PACKAGE (\$30.00)

* SC.XREF - Generates a GLOBAL LABEL Cross Reference Table for complete documentation of source listings. Formatting control accommodates all printer widths for best hardcopy outputs. * SC.GSR - Global Search and Replace eliminates tedious manual renaming of labels. Search all or part of source. Optional prompting for user verification. * SC.TAB - Tabulates source files into neat, readable form. **SOURCE code: \$40.00**

HARDWARE/FIRMWARE**THE 'PERFORMER' CARD (\$39.00)**

Plugs into any Apple slot to convert your 'dumb' centronics-type printer I/F card into a 'smart' one. Command menu provides easy access to printer fonts. Eliminates need to remember complicated ESC codes and key them in to setup printer. Added features include perforation skip, auto page numbering with date & title. Also includes large HIRES graphics screen dump in normal or inverse plus full page TEXT screen dump. Specify printer: Epson MX-80 with Graftrax-80, MX-100, MX-80/100 with GraftraxPlus, NEC 80923A, C.Itoh 8510 (Prowriter), OkiData 82A/83A with Okigraph & OkiData 92/93. Oki bonus: print EMPHASIZED & DOUBLE STRIKE fonts! **SOURCE code: \$30.00**

FIRMWARE FOR APPLE-CAT: The 'MIRROR' ROM (\$25.00)

Communications ROM plugs directly into Novation's Apple-Cat Modem card. Three basic modes: Dumb Terminal, Remote Console & Programmable Modem. Added features include: selectable pulse or tone dialing, true dialtone detection, audible ring detect, ring-back option and built-in printer buffer. Supports most 80-column displays and the l-wire shift key mod. Uses a superset of Apple's Comm card and Micromodem II commands. A-C hardware differences prevent 100% compatibility with Comm card. **SOURCE code: \$60.00**

RAM/ROM DEVELOPMENT BOARD (\$30.00)

Plugs into any Apple slot. Holds one user-supplied 2Kx8 memory chip. Use a 6116 type RAM chip for program development or just extra memory. Plug in a preprogrammed 2716 EPROM to keep your favorite routines 'on-line'. A versatile board with many uses! Maps into \$Cn00-\$nFF and \$CB00-\$FFFF memory space. Circuit diagram included.

NEW!!! SINGLE BOARD COMPUTER KIT (\$20.00)

Kit includes etched PC board (with solder mask and plated thru holes) and assembly instructions. User provides 6502 CPU, 6116 2K RAM, 6821 dual 8-bit I/O and 2732 4K EPROM plus misc common parts. Originally designed as intelligent printer interface - easily adapted to many applications needing dedicated controller. (Assembled and tested: \$119.00)

All assembly language SOURCE code is fully commented & provided in both S-C Assembler & standard TEXT formats on an Apple DOS 3.3 diskette. Specify your system configuration with order. Avoid a \$3.00 postage and handling charge by enclosing full payment with order (MasterCard & VISA excluded). Ask about our products for the VIC-20 and Commodore 64!

R A K - W A R E

41 Ralph Road West Orange NJ 07052 (201) 325-1885

```

1340 *-----*
1350 *
1360 *      ADDRESS TABLE
1370 *
000F- 1380 ODDMSK .EQ $FO      MASK FOR ELIMINATING UPPER BLOCK
000F- 1390 EVNMSK .EQ $OF      MASK FOR ELIMINATING LOWER BLOCK
FB40- 1400 GRAPH .EQ $FB40    ENABLE LO RES GRAPHICS
1410 *
1420 *      MACRO DEFINITIONS
1430 *
1440     .MA PLY      PLY ]1
1450     LDY ]1      Y-COORD
1460     LDA LORES,Y
1470     STA POINTR
1480     LDA LORESH,Y
1490     STA POINTR+1
1500     TYA
1510     AND #1      GET LSB
1520     TAX
1530     .EM
1540 *
1550     .MA PLX      PLX ]1
1560     LDY ]1      X-COORD
1570     LDA (POINTR),Y GET THE PAGE BYTE
1580     AND MASK,X
1590     ORA COLOR,X
1600     STA (POINTR),Y PUT IT BACK
1610     .EM
1620 *
1630     .MA NEXT     NEXT VAR LIMIT+1
1640     INC ]1      INCREMENT ]1 VARIABLE
1650     LDA ]1      TEST IF ]1=]2 YET
1660     CMP ]2
1670     BCS :1      YES, LEAVE LOOP
1680     JMP NEXT.]1 NO, KEEP LOOPING
1690 :1
1700     .EM
1710 *
1720     .MA GET      GET FORMULA, INDEX
1730     LDY I
1740     LDA ]1L-1,Y
1750     STA POINTR
1760     LDA ]1H-1,Y
1770     STA POINTR+1
1780     LDY ]2
1790     LDA (POINTR),Y
1800     .EM
1810 *
1820 *
1830 *
1840 *      MAIN LOOP
1850 *
6000- AD 56 CO 1860 ROD    LDA $C056    SET LORES GRAPHICS ON
6003- AD 53 CO 1870         LDA $C053    MIXED MODE
6006- 20 H0 FB 1880 JSR GRAPH
6009- A9 F0 1890 LDA #ODDMSK
600B- 85 0A 1900 STA MSKODD
600D- A9 OF 1910 LDA #EVNMSK
600F- 85 OB 1920 STA MSKEVN
1930 *
1940 *      BIG LOOP
6011- 20 E2 FB 1950 JSR $FBEE2   *** TESTING BEEP ***
6014- A9 00 1960 LDA #0      FOR W=3 TO 50 (0...47)
6016- 85 FE 1970 STA W
1980 *----NEXT W COMES HERE-----
6018- A9 01 1990 NEXT.W LDA #1      FOR I=1 TO 19
601A- 85 FB 2000 STA I
601C- A9 27 2010 LDA #39
601E- 85 EE 2020 STA INV1
6020- AD 30 CO 2030 LDA $C030    JUST FOR AUDIBLE FEEDBACK
2040 *----NEXT I COMES HERE-----
6023- A5 FB 2050 NEXT.I LDA I      SET UP K = I+J
6025- 85 FD 2060 STA K
6027- A5 EE 2070 LDA INV1
6029- 85 EF 2080 STA INVK
602B- 85 07 2090 >GET FORM1,W
603B- 85 07 2100 STA COLOR1  SAVE IT FOR INNER LOOP
603D- A9 00 2110 LDA #0      FOR J=0 TO 19
603F- 85 FC 2120 STA J

```

```

2130 *---NEXT J COMES HERE-----
2140 NEXT.J >GET FORM2,J
2150 CLC ADD THE FORMULAS
2160 ADC COLOR1 ACC = J*3/(I+3)+I*W/12
2170 AND #$OF MASK OFF TOP
2180 STA COLEVN EVEN COLOR
2190 ASL SHIFT 4 BITS
2200 ASL
2210 ASL
2220 ASL
2230 STA COLODD ODD COLOR
2240 *
2250 >PLY I
2260 >PLX K
2270 >PLX INVK
2280 *
2290 >PLY INVI
2300 >PLX K
2310 >PLX INVK
2320 *
2330 >PLY K
2340 >PLX I
2350 >PLX INVI
2360 *
2370 >PLY INVK
2380 >PLX I
2390 >PLX INVI
2400 *
2410 INC K
2420 DEC INVK
2430 >NEXT J,20
2440 *
2450 DEC INVI
2460 >NEXT I,20
2470 *
2480 >NEXT W,48
2490 *
6041- 18
6051- 65 07
6052- 29 OF
6054- 85 08
6056- 0A
6058- 0A
6059- 0A
605A- 0A
605B- 0A
605C- 85 09
605E-
606E-
6078-
6082-
6092-
609C-
60A6-
60B6-
60C0-
60CA-
60DA-
60E4-
60EE- E6 FD
60FO- C6 EF
60F2-
60FD- C6 EE
60FF-
610A-
6115- AD 00 CO
6118- 30 03
611A- 4C 11 60
611D- 8D 10 CO
6120- 60
2500 LDA $C000 ANY KEY PRESSED?
2510 BMI ROD4 YES
2520 JMP BIG LOOP NO, KEEP LOOPING
2530 ROD4 STA $C010
2540 RTS

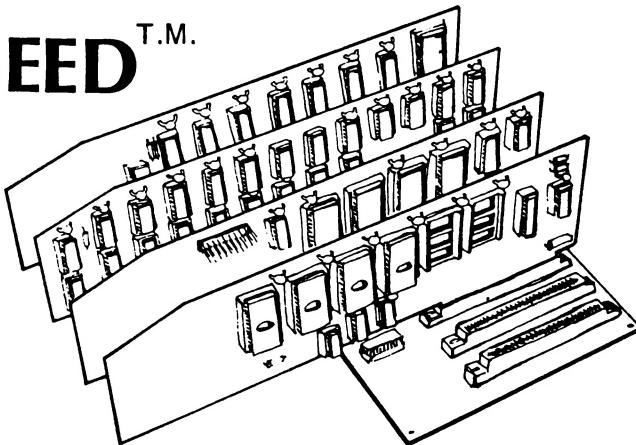
```

```

2550 *
2560 *
2570 * LORES GRAPHICS PAGE
2580 * BASE ADDRESSES (40 COL)
2590 *
2600 LORESL .HS 0000808000008080
2610 .HS 0000808000008080
2620 .HS 2828A8A82828A8A8
2630 .HS 2828A8A82828A8A8
2640 .HS 5050D0D05050D0D0
2650 LORESH .HS 0404040405050505
2660 .HS 0606060607070707
2670 .HS 0404040405050505
2680 .HS 0606060607070707
2690 .HS 0404040405050505
2700 *
2710 *
2720 *
2730 * TABLE FOR I*W/12
2740 *
2750 FORM1L .DA #WI1,#WI2,#WI3,#WI4,#WI5,#WI6,#WI7
2760 .DA #WI8,#WI9,#WI10,#WI11,#WI12,#WI13,#WI14
2770 .DA #WI15,#WI16,#WI17,#WI18,#WI19
2780 FORM1H .DA /WI1,/WI2,/WI3,/WI4,/WI5,/WI6,/WI7
2790 .DA /WI8,/WI9,/WI10,/WI11,/WI12,/WI13,/WI14
2800 .DA /WI15,/WI16,/WI17,/WI18,/WI19

```


APPLESEED^{T.M.}



Appleseed is a complete computer system. It is designed using the bus conventions established by Apple Computer for the Apple][. Appleseed is designed as an alternative to using a full Apple][computer system. The Appleseed product line includes more than a dozen items including CPU, RAM, EPROM, UART, UNIVERSAL Boards as well as a number of other compatible items. This ad will highlight the Mother board.

BX-DE-12 MOTHER BOARD

The BX-DE-12 Mother board is designed to be fully compatible with all of the Apple conventions. Ten card slots are provided. Seven of the slots are numbered in conformance with Apple standards. The additional three slots, lettered A, B and C, are used for boards which don't require a specific slot number. The CPU, RAM and EPROM boards are often placed in the slots A, B and C.

The main emphasis of the Appleseed system is illustrated by the Mother Board. The absolute minimum amount of circuitry is placed on the Mother Board; only the four ICs which are required for card slot selection are on the mother board. This approach helps in packaging (flexibility & smaller size), cost (buy only what you need) and repairability (isolate and fix problems through board substitution).

Appleseed products are made for O.E.M.s and serious industrial/scientific users. Send for literature on the full line of Appleseed products; and, watch here, each month, for additional items in the Appleseed line.

Appleseed products are not sold through computer stores.

Order direct from our plant in California.

Apple is a registered trademark of Apple Computer, Inc.

DOUGLAS ELECTRONICS

718 Marina Blvd., San Leandro, CA 94577 • (415) 483-8770

Will ProDOS Really Fly?.....Bob Sander-Cederlof

ProDOS appears to have been eclipsed by MacIntosh. The major software houses are probably putting their main effort into Mac.

ARTSCI has announced a ProDOS version of their MagiCalc spreadsheet program. Owners of the DOS 3.3 version may upgrade for \$40, new customers pay \$149.95. The only differences claimed are faster disk I/O and ability to edit the printer setup string. Nice, but \$40 is a lot. And the spreadsheet files would no longer be accessible to DOS-based utilities.

ARTSCI will also send you their ProDOS catalog sorter program, complete with BASIC.SYSTEM, CONVERT, FILER, and the ProDOS image for only \$24.95. Apple will reputedly be selling ProDOS with a user's manual and some tutorial files in addition to the files on ARTSCI's disk, but price and date are still unclear. (You get them free with a new disk drive.)

Practical Peripherals has announced a new clock card which is ProDOS compatible. Their design is apparently an upgrade of Superclock II (by Jeff Mazur, Westside Electronics). ProDOS was designed around Thunderclock, so other clocks must either emulate one of the Thunderclock modes or patch ProDOS during the boot process. Applied Engineering's new Timemaster II emulates Thunderclock and several others, so it is fully ProDOS compatible.

According to Don Lancaster, Applewriter //e has been written so that changing to ProDOS would be easy. Therefore we might expect a ProDOS-based version of this popular word processor to be announced soon. Or maybe they won't bother to announce it.

Meanwhile, I know of at least two people with plans to integrate the faster RWTS ProDOS uses into their enhanced DOS 3.3 packages. Have you seen the latest ads for David-DOS? Dave Weston compares the speeds of his fast DOS with DOS 3.3 and Pro-DOS. Guess what ... Pro-DOS doesn't win.

Unless you MUST have file compatibility with Apple /// SOS; or you MUST have hard hard-disk support for very large files; or you MUST have a hierarchical file directory; then stick with DOS 3.3, enhanced by Dave, or Bill Basham, or Art Schumer, or others. And if you MUST have at least 32K of program space with Applesoft; or you MUST have Integer BASIC support; or you MUST have compatibility with hundreds of existing software products; then stick with DOS 3.3.

Apple Assembly Line is published monthly by S-C SOFTWARE CORPORATION, P.O. Box 280300, Dallas, Texas 75228. Phone (214) 324-2050. Subscription rate is \$18 per year in the USA, sent Bulk Mail; add \$3 for First Class postage in USA, Canada, and Mexico; add \$12 postage for other countries. Back issues are available for \$1.50 each (other countries add \$1 per back issue for postage).

All material herein is copyrighted by S-C SOFTWARE CORPORATION, all rights reserved. (Apple is a registered trademark of Apple Computer, Inc.)